



IBM Power Systems

# SMT Verification of the POWER5 and POWER6 High-Performance Processors

**John Ludden**

Senior Technical Staff Member

Hardware Verification

IBM Systems & Technology Group

# Introduction to Simultaneous Multi-Threading (SMT)

## 1. What is a multi-threaded processor?

- **Essentially a processor core that executes multiple instruction streams simultaneously**
- **Each thread appears to software as a “virtual” processor core**

## 2. What are the advantages of SMT?

- **More efficient utilization of silicon real estate and power: small die size increase compared to adding another core**
- **Increased system throughput by utilizing processor resources that would otherwise be idle**

## 3. What are the disadvantages of SMT?

- **Increased complexity -> Makes verification state space MUCH larger**
  - **SMT verification much harder than SMP**
- **Possibly degrades performance of some applications**

# Examples of SMT microprocessors

## 1. Video Game Systems

- **Sony Playstation 3: IBM CELL processor**
- **Xbox 360: IBM Xenon processor**

## 2. Personal Computers:

- **Intel Pentium 4 Hyper-Threading (HT) processors**

## 3. Servers:

- **SUN UltraSparc Systems: T1 (4 threads) and T2 (8 threads)**
- **HP Superdome Systems: Intel Itanium 2**
- **IBM Power Systems: POWER5 and POWER6 processors**

# Overview

- 1. Context : POWER5 vs. POWER6 Microarchitecture Comparison**
- 2. Verification methodology: In the beginning...**
- 3. The times they are a changing: SMT arrives in POWER5**
- 4. POWER6: An in-order design should be simpler, but...**
- 5. Future directions?**

# IBM POWER systems

**Consistent predictable delivery**



2001

**POWER4**



2003

**POWER4+**



2004

**POWER5**



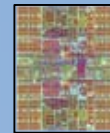
2006

**POWER5+**

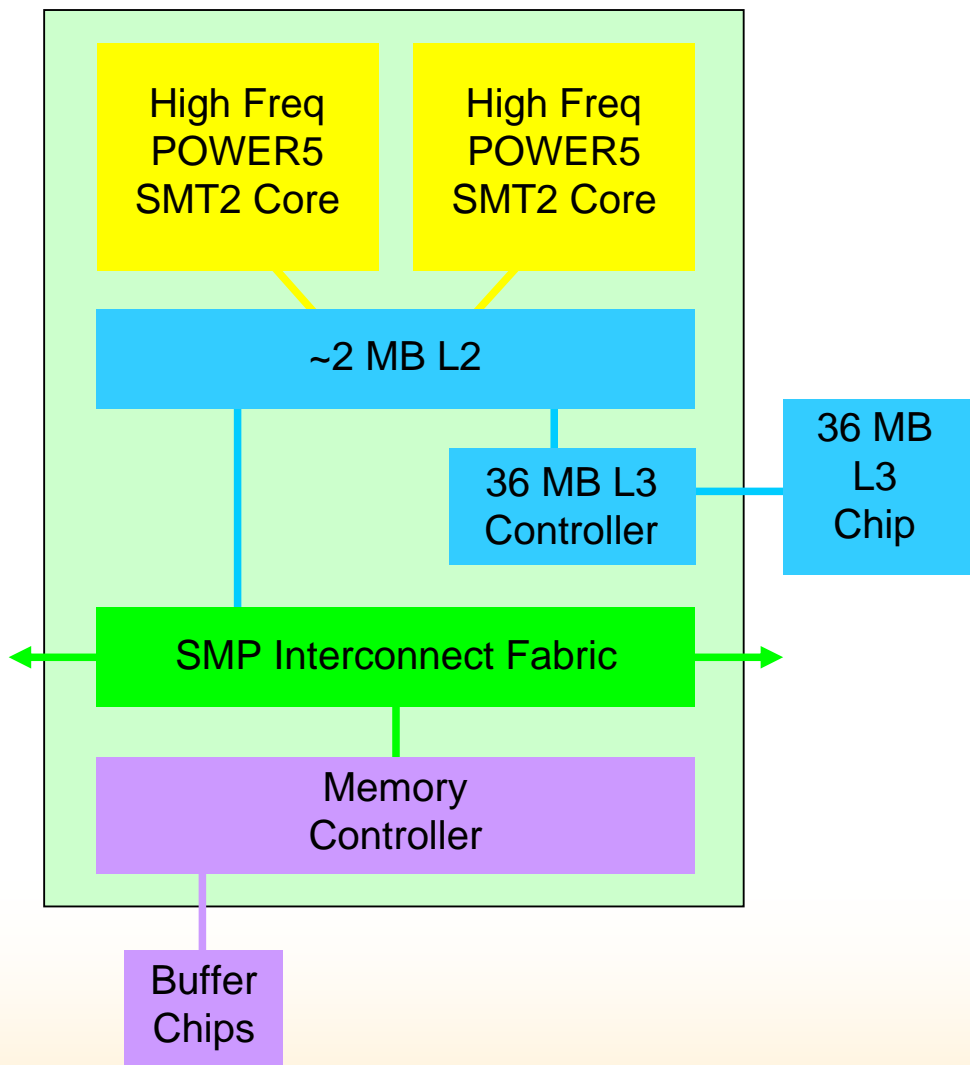


2007

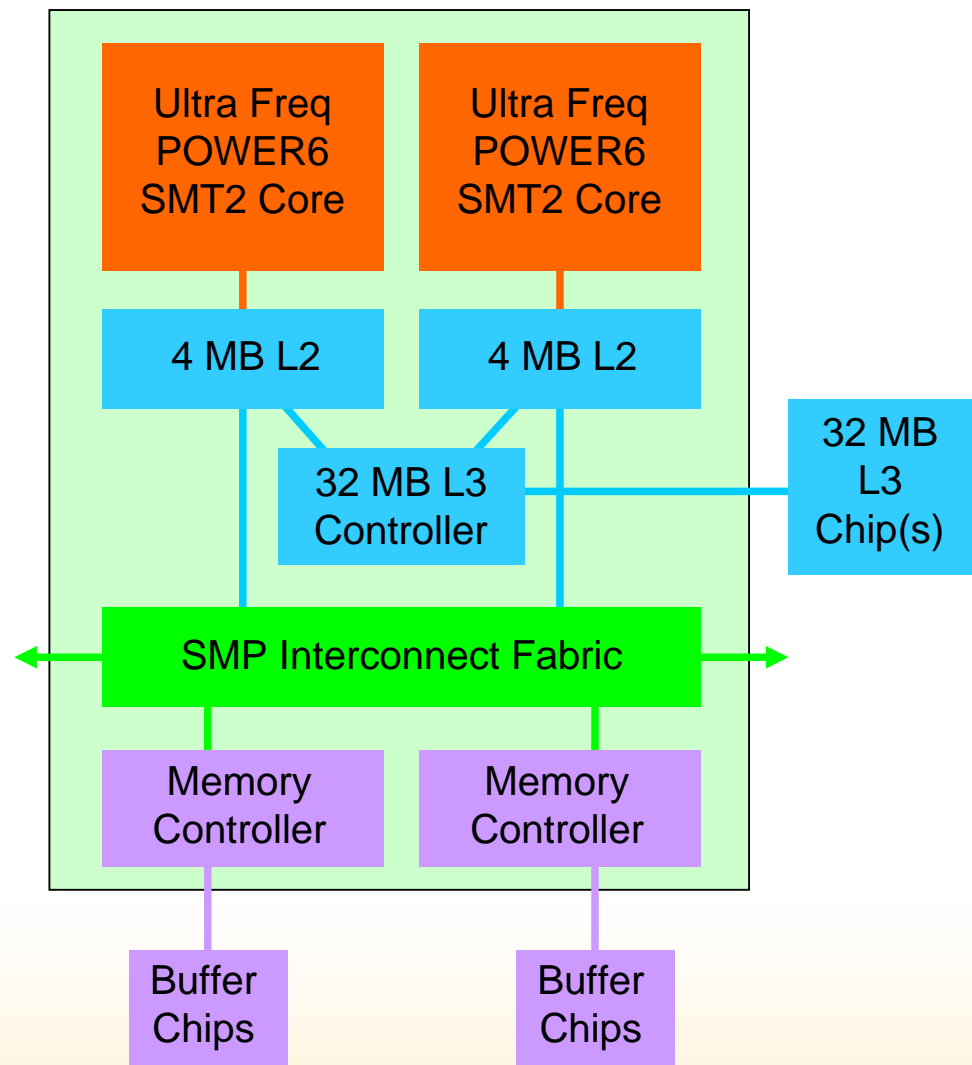
**POWER6**



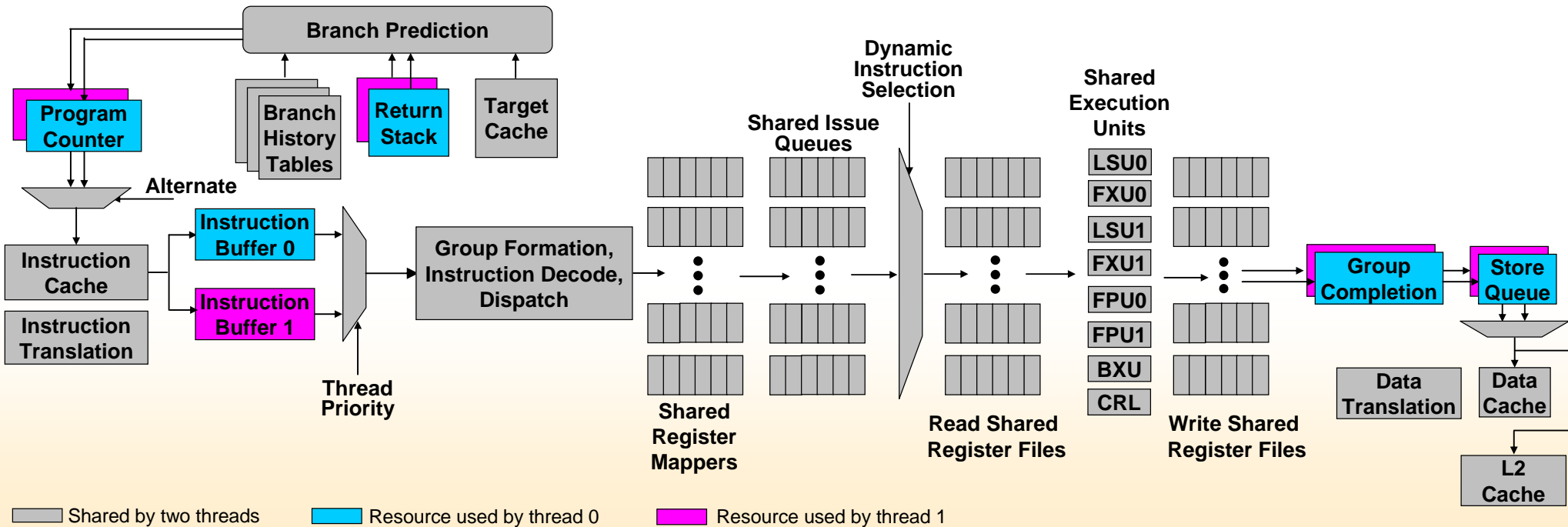
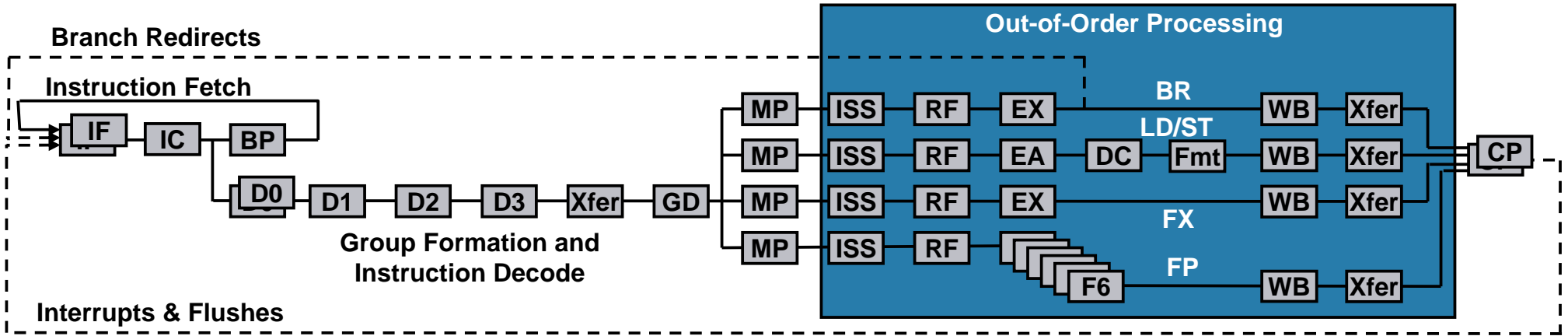
### POWER5 Chip



### POWER6 Chip



# POWER5 Pipeline



# High-end server: New POWER6 microprocessor

## ▪ Topology

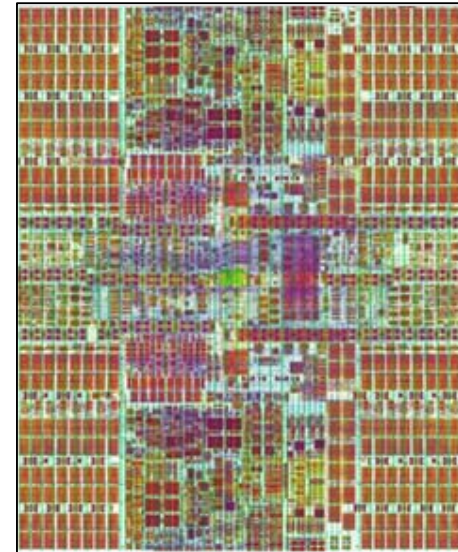
- Two cores on chip, a 2-way SMP
- Core private L1s (64KB I, 64KB D)
- Superscalar, SMT cores
- Chip private 8 MB L2 cache
- L3 32 MB off chip
- Two-tier SMP fabric

## ▪ Technology

- 65 nm SOI
- 341 mm<sup>2</sup> die size
- 10 Layers of metal
- 790 million transistors on chip
- Frequency : 3.5, 4.2, 4.7, 5.0 GHz

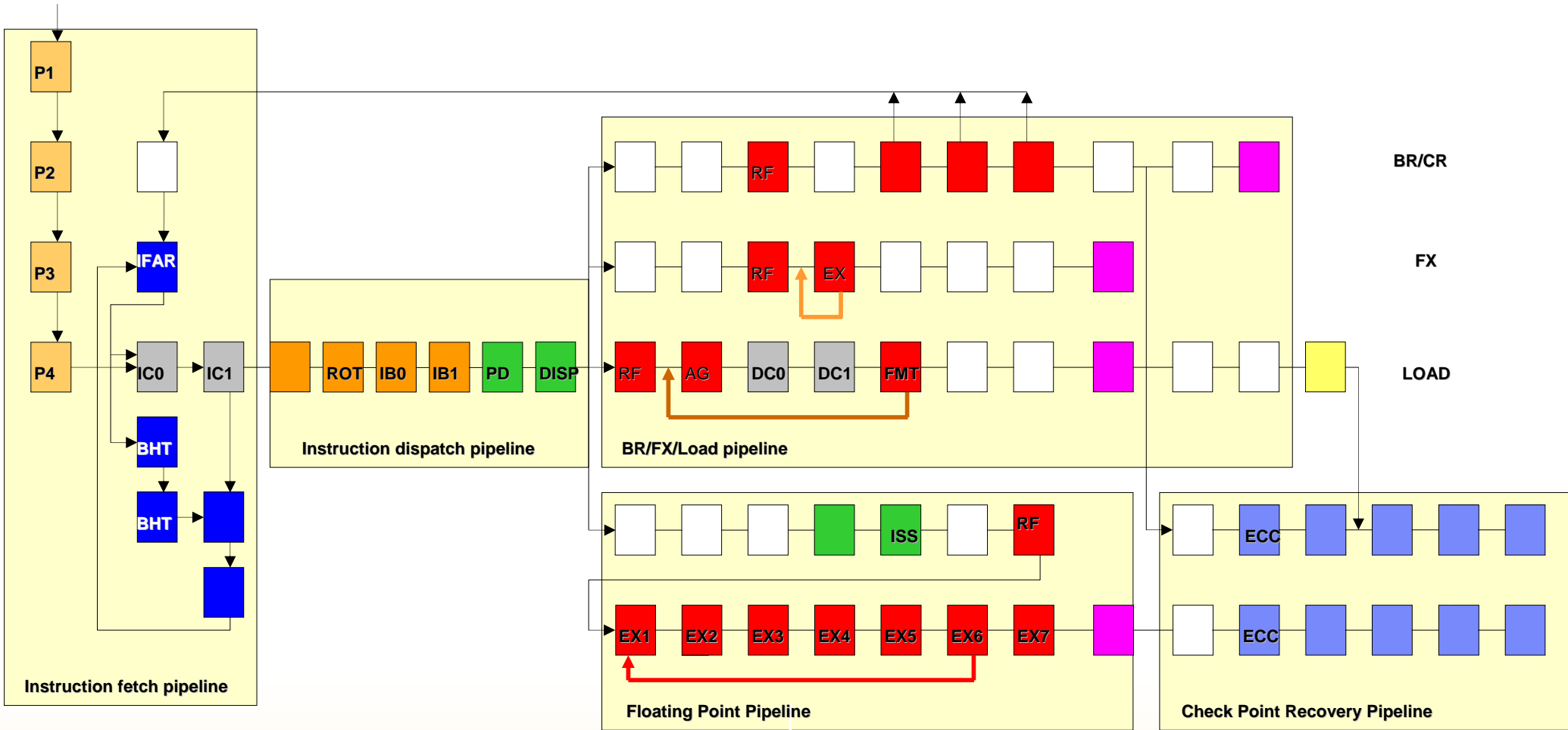
## ▪ Custom & semi-custom design style

- High frequency constraints



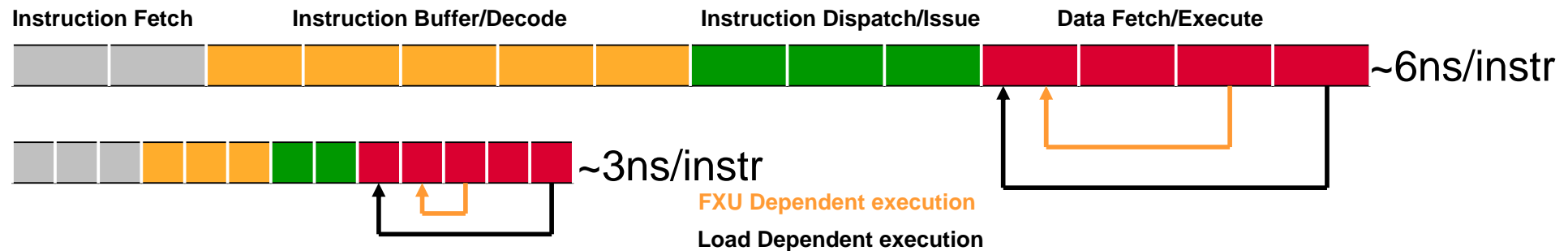
**3.3 M Lines of VHDL**

# POWER6 core pipeline



# POWER6 core

- **POWER6 processor is ~2X frequency of POWER5 (4 – 5 GHz)**
- **POWER6 instruction pipeline depth equivalent to POWER5**
  - Minimize power
  - Scale performance with frequency

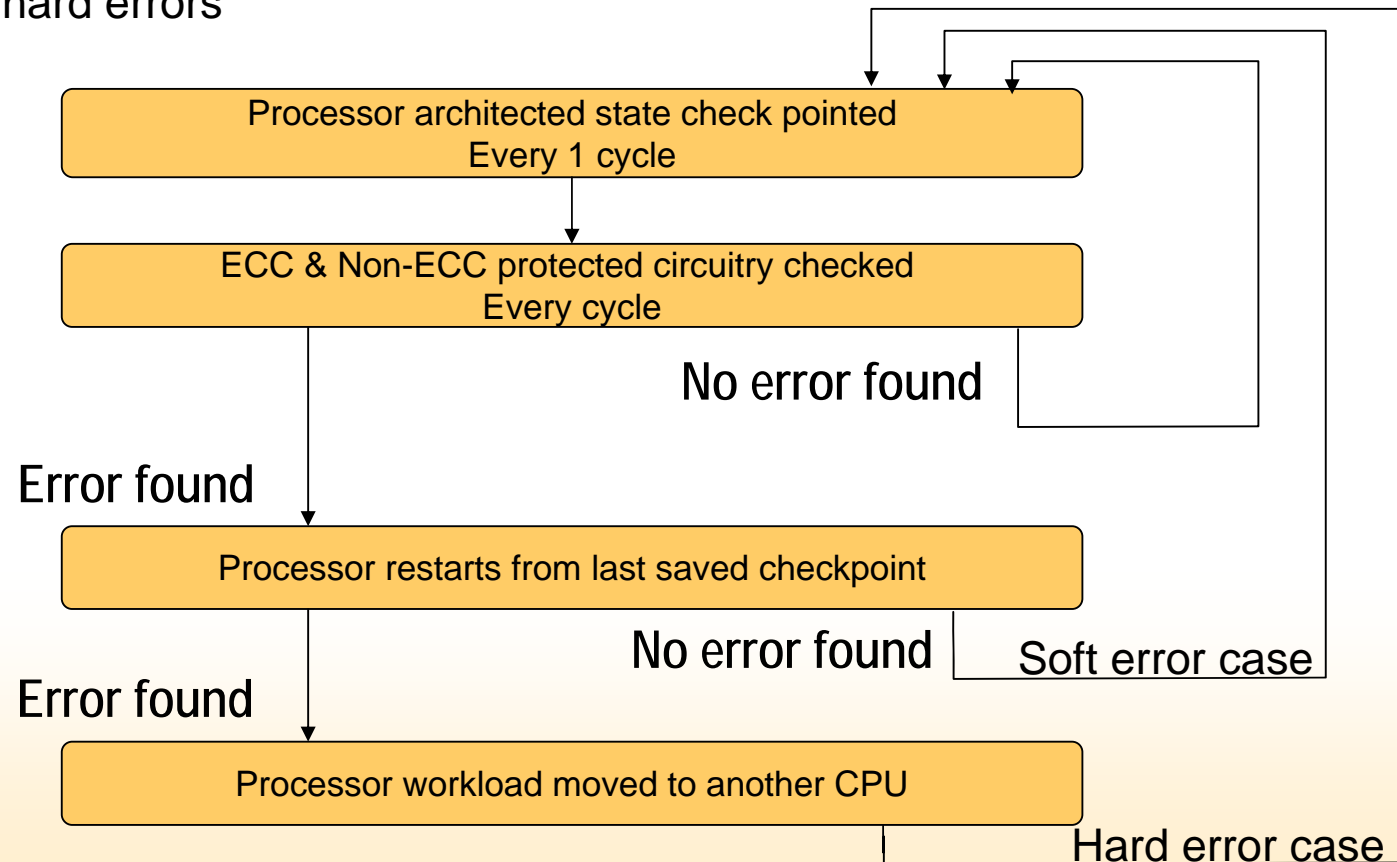


- **POWER6 extends functionality of POWER5 core**
  - 64K I cache, 64K D cache, 2 FXU, 2 Binary FPU, 1 branch execution unit
  - **Two way SMT with 7 instruction dispatch from 2 threads (maximum of 5 instructions per thread)**
  - Decimal Floating Point Unit
  - VMX Unit (PowerPC's SIMD ISA)
  - Recovery Unit

# Bullet-proof computing

## ▪ System reliability with recovery unit

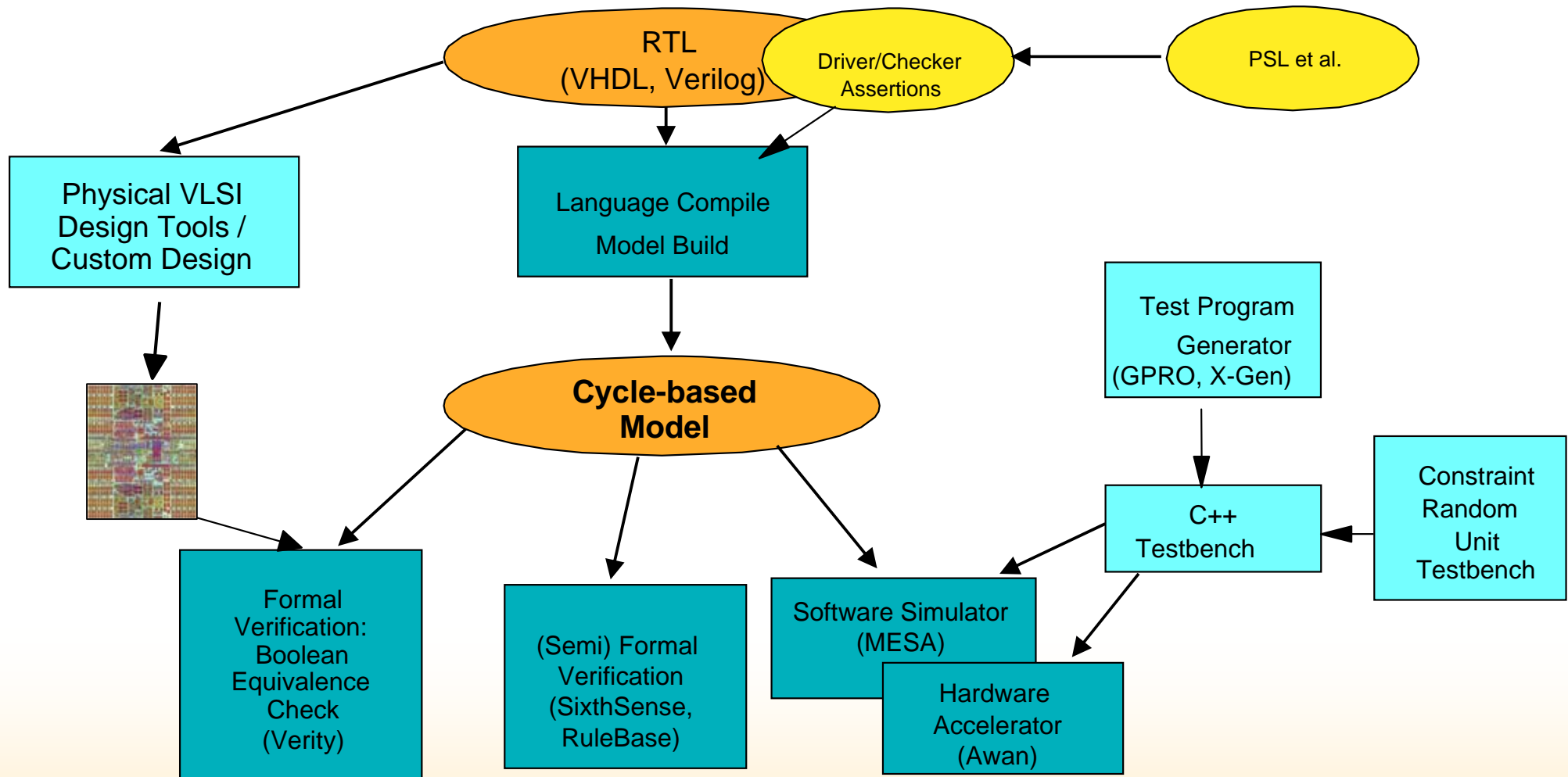
- Every measure possible taken to preserve application execution
- Retry soft errors
- Change hardware for hard errors



# Overview

- 1. Context : POWER5 vs. POWER6 microarchitecture comparison**
- 2. Verification methodology: In the beginning...**
- 3. The times they are a changing: SMT arrives in POWER5**
- 4. POWER6: An in-order design should be simpler, but...**
- 5. Future directions?**

# POWER4/5/6 RTL verification technology



# Single threaded uniprocessor verification for POWER4

## ▪ **Unit level: methodology inherited from POWER4**

- Driven by a combination of instruction level test cases (AVPs) created by Genesys-Pro (GPRO) pseudo-random test generator and random C++ driven irritation
- Instruction-By-Instruction (IBI) checking against AVP results
- Low level microarchitecture checkers written in C++

## ▪ **Processor core (aka “core”) level**

- Mixture of GPRO pseudo-random and directed random instruction level test cases
- IBI checking against AVP results
- Low level microarchitecture checkers written in C++
- Irritation from random C++ drivers
- Highly deterministic and architected state easily verifiable against test

# Symmetric multi-processor (SMP) verification for POWER4

## Chip (dual-core) level

- Test generation similar to uniprocessor via GPRO for false-sharing or non-sharing tests
  - IBI checking against AVP results for two-independent instruction streams contained within single test
  - Low level microarchitecture checkers written in C++
  - L1/L2 interactions primary focus
  
- True-sharing scenarios, lock testing and storage access (“weak”) ordering checked
  - GPRO employed but....
    - IBI checking of these accesses is limited or not possible:
      - › Non-unique or non-deterministic results
      - › CML (architecture level coherency monitor) employed to detect the “right answer” as a post-simulation rule check

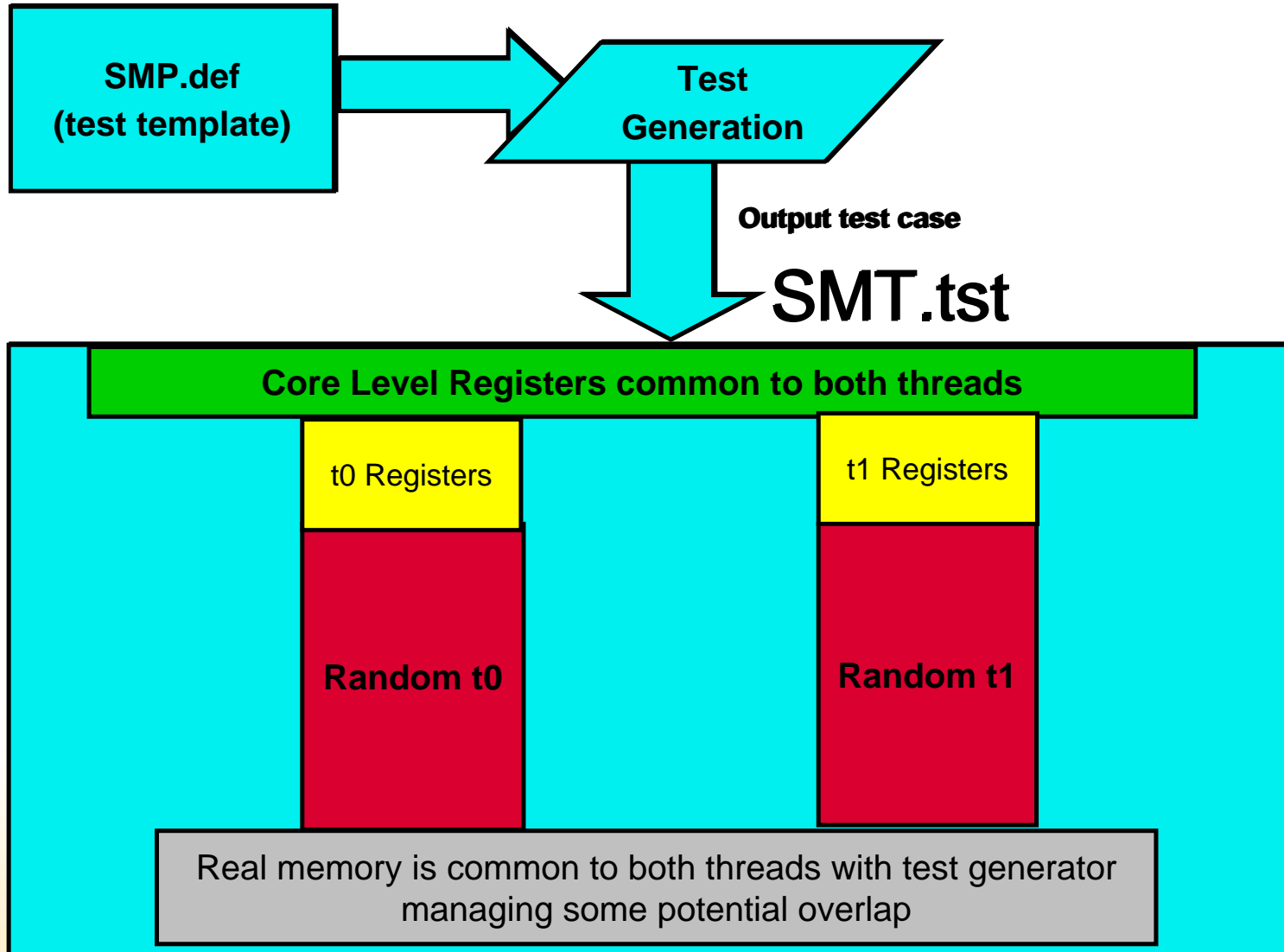
# Overview

1. **Context : POWER5 vs. POWER6 microarchitecture comparison**
2. **Verification methodology: In the beginning...**
3. **The times they are a changing: SMT arrives in POWER5**
4. **POWER6: An in-order design should be simpler, but...**
5. **Future directions?**

# POWER5 SMT verification methodology

- **Evolutionary based on single thread uniprocessor and SMP approaches**
  - Traditional SMP scenarios now self-contained in a single core simulation model
    - Downward migration of dual-core methodology to single core model
  
- **New SMT verification scenario categories**
  - Shared resource and priority conflicts:
    - SMT resource types:
      - Equally shared between threads: Queue full conditions easier to hit
      - Dynamically shared / tagged: Either thread can consume most/all of the resource
      - Replicated: Not shared...same as single thread
  - Dynamic thread mode switching: SMT->ST; ST->SMT
    - Some applications attain better performance in ST mode
    - Shared resources re-allocated on each mode switch

# Traditional SMP approach applied to SMT verification

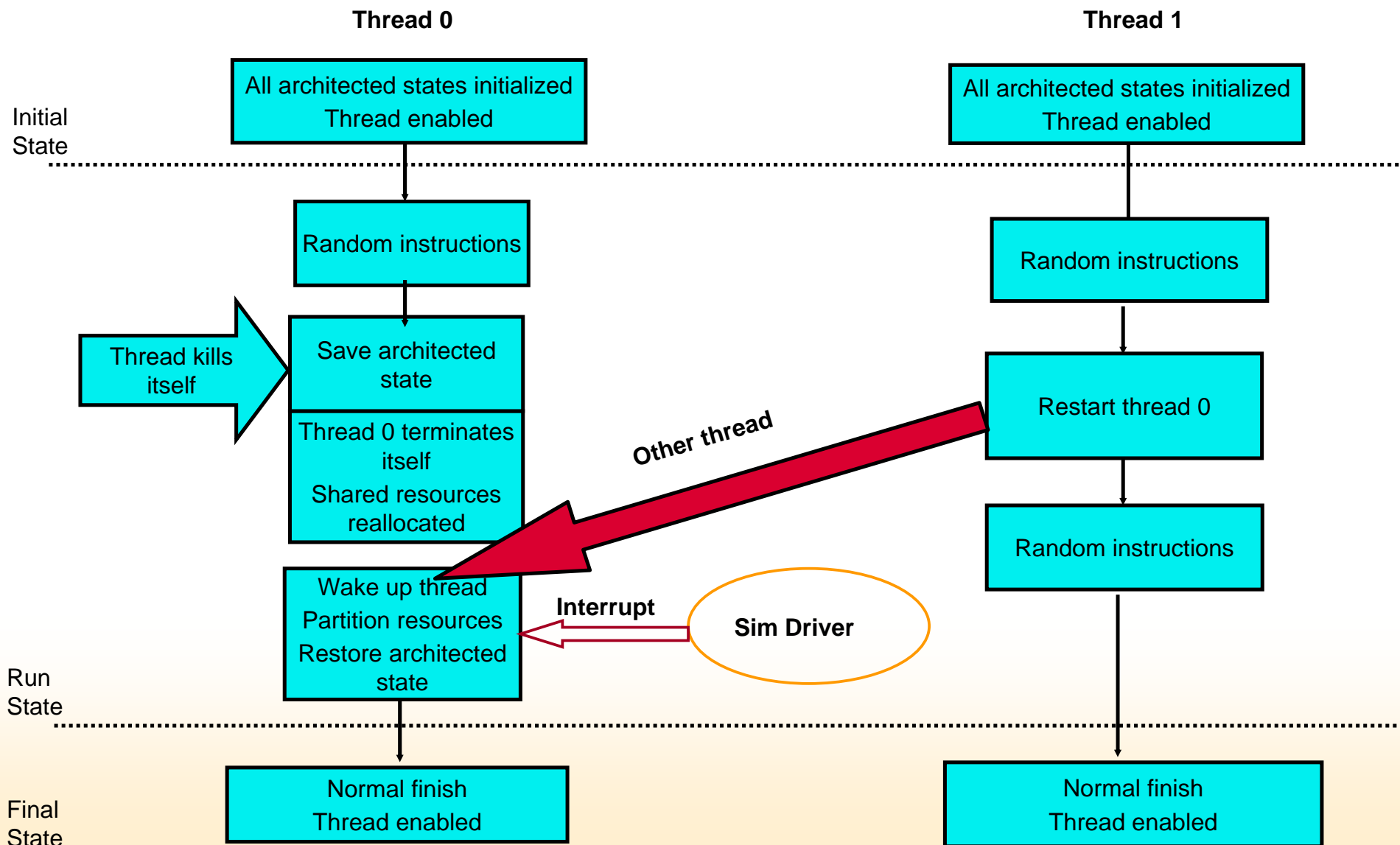


# Shared resource and priority conflicts

## Approach was similar to SMP verification

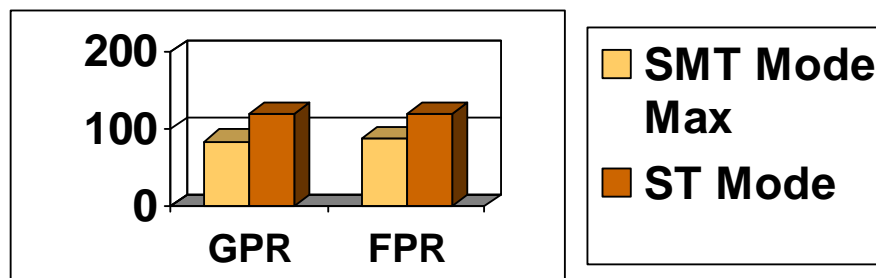
- Testing largely consisted of “symmetric” instruction streams on each thread
  - A particular resource targeted (e.g., GPR rename registers)
    - 100 load instructions on each thread
- Coverage and lab feedback validated this approach
  - Good enough: “Got the job done”

# POWER5 dynamic thread mode switching

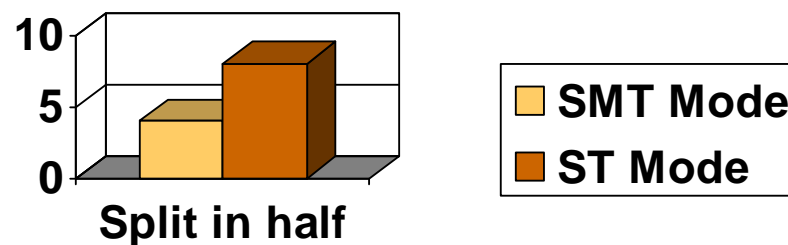


# POWER5 shared resource re-allocation on mode switch

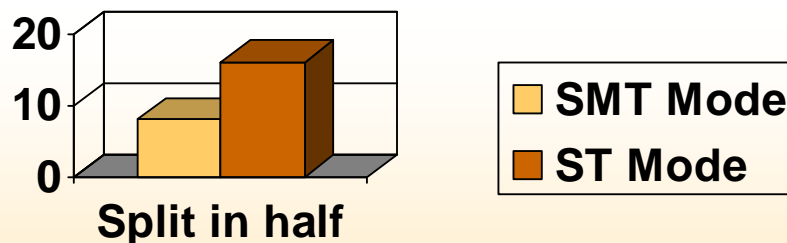
### Rename Registers per thread



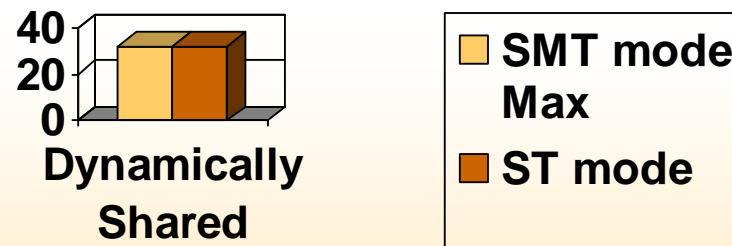
### Load Miss Queue entries per thread



### Branch Queue (BIQ) entries per thread



### Max LRQ/SRQ entries per thread



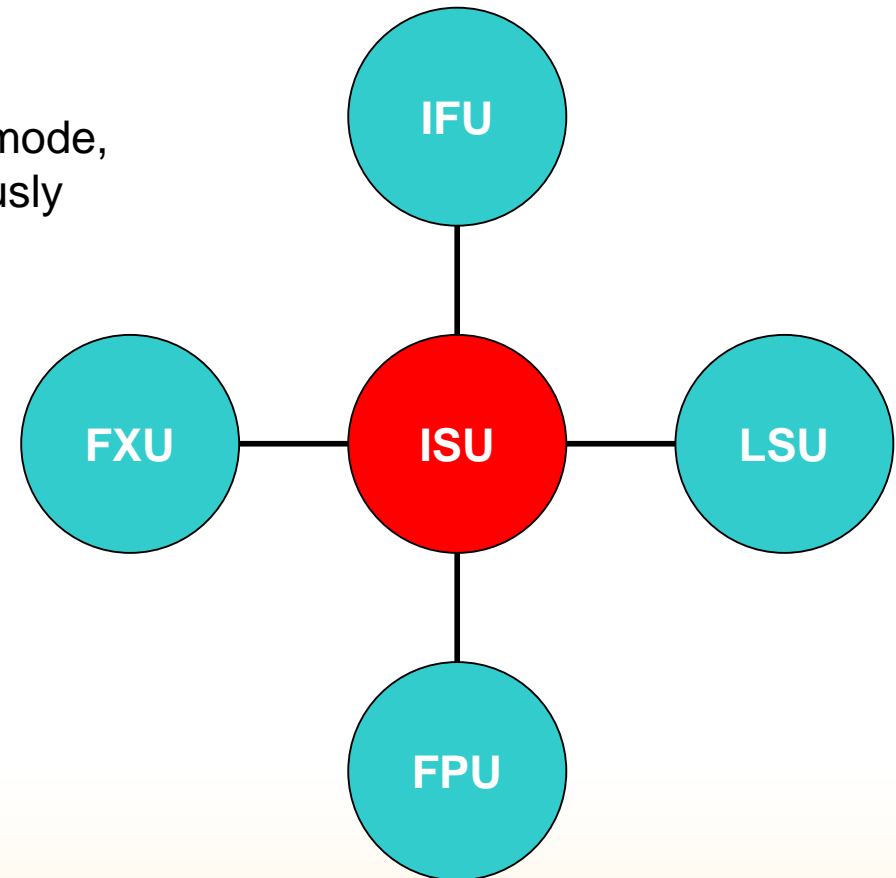
# Overview

1. **Context : POWER5 vs. POWER6 microarchitecture comparison**
2. **Verification methodology: In the beginning...**
3. **The times they are a changing: SMT arrives in POWER5**
4. **POWER6: An in-order design should be simpler, but...**
5. **Future directions?**

# POWER5: centralized complexity

## POWER5

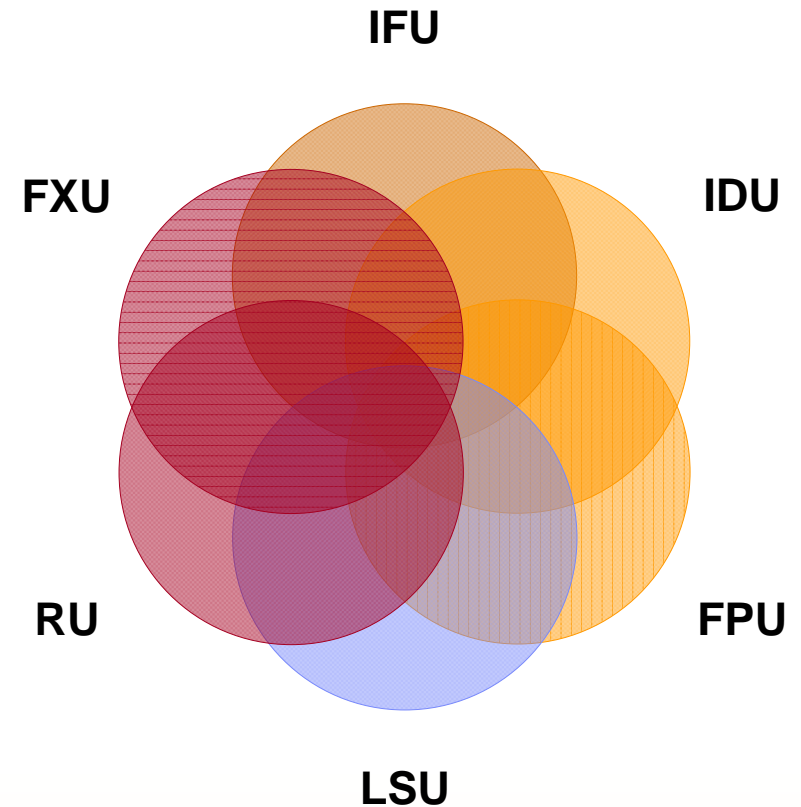
- Out-of-order design: Even in single thread mode, complex events naturally occur simultaneously
- Started from POWER4+: Known working design that was modified incrementally
- 23 FO4 design: Isolated complexity in Instruction Sequencing Unit (ISU):
  - Every unit communicated back to ISU
  - ISU resolved all exceptions and out-of-order conflicts
- ST and SMT modes both supported:
  - Alternating dispatch cycles per thread
  - Resources re-allocated on mode switch



# POWER6 distributed complexity

## POWER6

- From-scratch mostly in-order design
  - Normally, design is well behaved
  - Cross-thread interaction necessary for “tough bugs”
- 13 FO4 design: Distributed complexity needed to achieve high performance goals
- Recovery unit (RU):
  - Must resolve out-of-order FP with in-order pipelines
  - Checkpoints machine state
  - Recovers processor from soft errors
- Design is inherently in SMT mode all the time (almost)
  - Dispatch to both threads in same cycle
  - Most resources dynamically shared / tagged
  - No resource reallocation on mode switch



# POWER6 verification process

## The different verification engines have different strengths related to the verification tasks

### ▪ Software simulation

- Slow, but low penalty for highly intrusive checking of model internals. Total model visibility.
- Hundreds of AIX workstations running 24x7x365
- New enhancements helped keep pace with design complexity
- 2x number of simulation cycles of POWER5 design

### ▪ Hardware-accelerated simulation

- 10-1k x Faster than SW sim, but need less intrusive driving/checking to not slow down hardware box.
- New usage: Mainline function verification
- Yields additional 3x simulation cycle advantage over POWER5 (5x cycle advantage overall)

### ▪ (Semi)-formal verification

- (High to) Exhaustive coverage, but higher skill needed to drive. Scaling problems w/ model size.
- Extensively used: Proved extremely valuable for complex SMT bugs

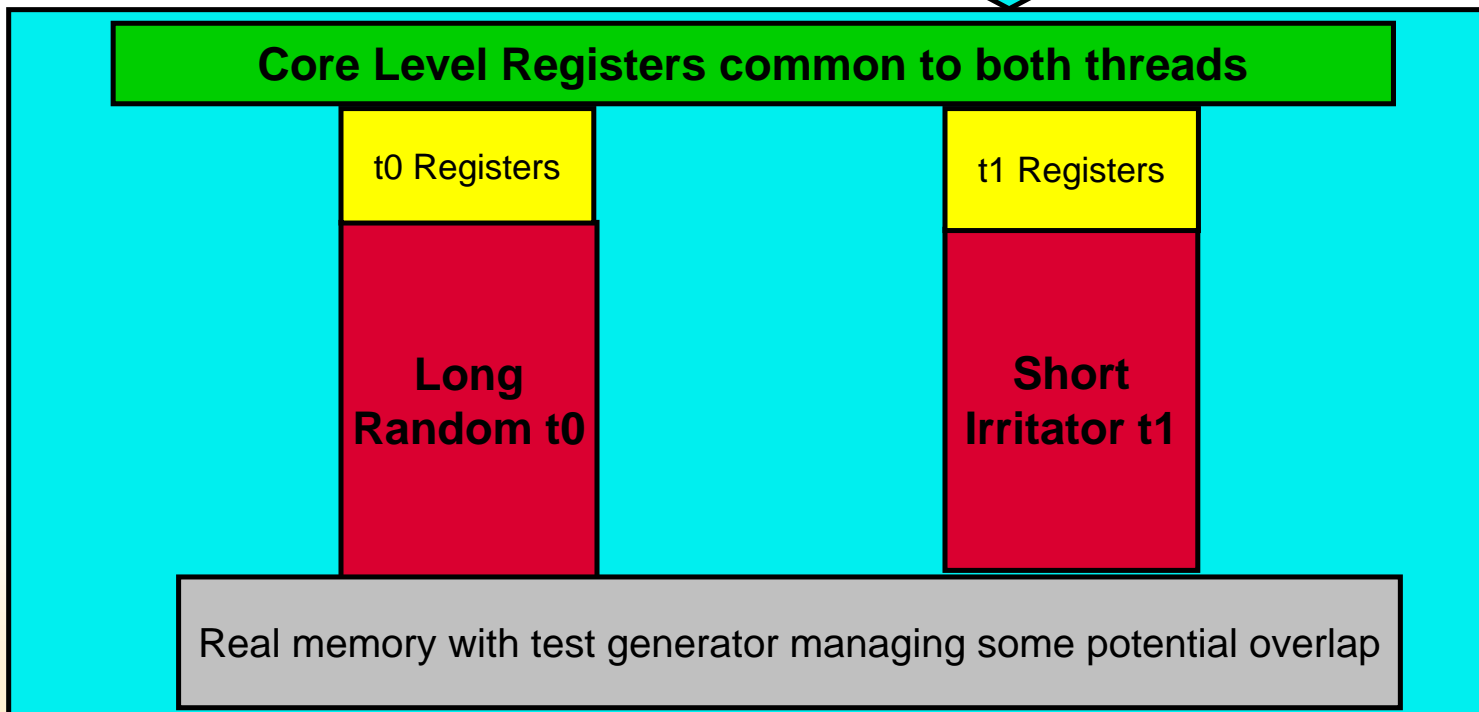
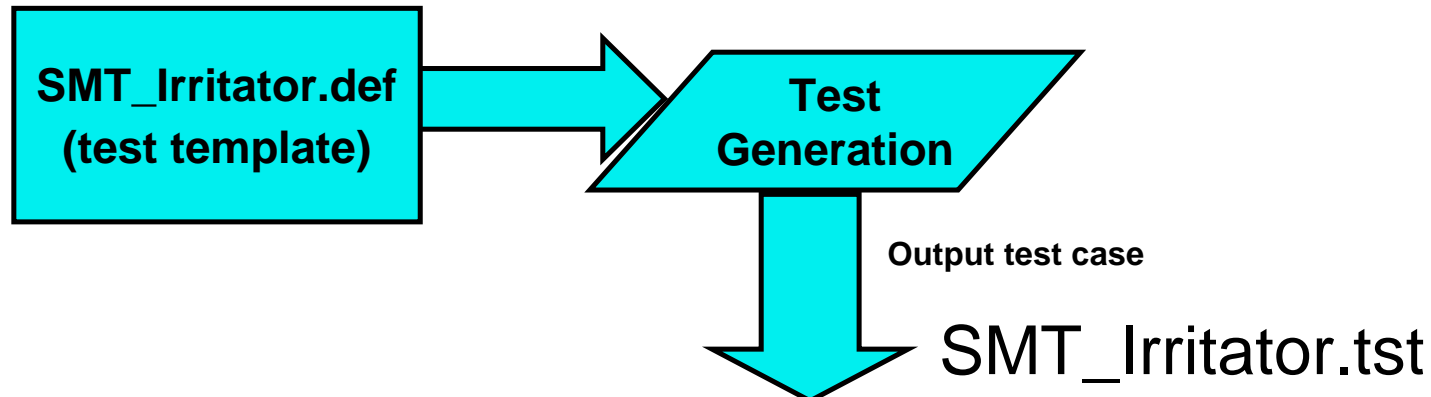
### ▪ Hardware bring-up

- Ideal speed, very limited visibility/controllability

# Software simulation enhancements

- **Random command driven unit simulation for most core units**
  - Yielded >1 Million lines of C++ code
  - More control over generation for low level events
  - More efficient test generation
  
- **Irritator threads at “core model” level**
  - “Symmetric” instruction stream approach employed on POWER5 proved inadequate
    - “S” in SMT is for “Simultaneous”, not “Symmetric”
  - Target cross-thread interactions at the microarchitecture level
  - ~2x test generation efficiency
  - Ensures both threads running the same length (self adjusting)

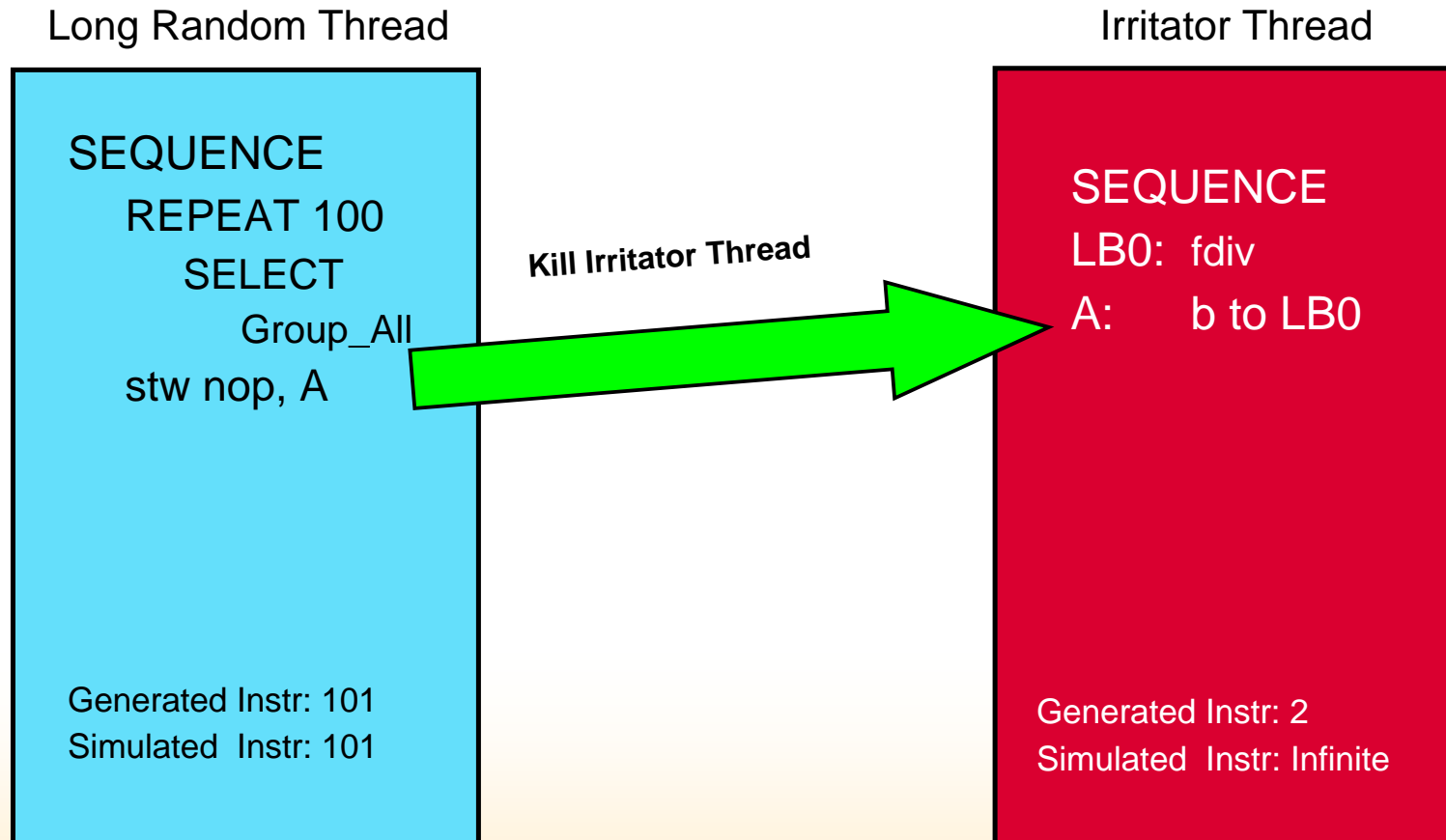
## Irritator thread example



## Irritator thread restrictions

- Cannot cause unexpected exceptions
- Cannot modify memory read by random thread
- Cannot modify registers shared with other threads
- Architected results may be undefined

# Irritator thread example



# Simulation acceleration usage on POWER6

## Extensively used on POWER6

- Run lab exercisers prior to tape-out
  - Found additional bugs missed by software simulation
  - Debug new exerciser functionality prior to lab
  - Error injection and recovery testing
  - Reproducibility of lab bugs in “simulation-like” environment for rapid debug of root cause
  - Rapid testing of bug fixes and collateral damage testing
- Linux boot prior to tape-out
- Not employed on POWER5 for “mainline” functional verification

## (Semi) Formal methods

### **Formal methods are a vital complement to simulation flow**

- Lab bring-up bug re-creation
  - Often faster reproduction than simulation based approaches
  - Aids in root cause analysis
  - High-coverage / proof of side-effect-free fixes

# Biggest challenge on POWER6

## Error detection and soft error recovery

- Why so hard?
  - Myriads of injection points coupled with large SMT state space
    - Often needed multiple “rare” combinations of “asymmetric” events on both threads while specific error was injected
  - End-to-end recovery testing difficult at unit level
    - Really a “core” effort
- Verification strategy:
  - Error injection and recovery on hardware accelerated simulation platform
  - Dynamic on-the-fly error injection combined with “irritator threads” needed to cover large SMT recovery state space

# Summary

- 1. SMT verification has four key pieces**
  - Traditional SMP-like effort
  - Thread starvation and priority
  - Starting and stopping threads
  - Asymmetric “irritator thread” approach to verify often unforeseen cross-thread interactions at the microarchitecture level
- 2. “From-scratch in-order” SMT design was more difficult to verify than the “out-of-order retrofitted” SMT design**
  - Complex events only occurred due to cross thread interaction
  - Even though team had experience
  - Required more “weapons” in the arsenal
- 3. High frequency design drove distributed complexity**
  - Makes verification job harder
  - Increased dependency on formal verification for difficult bugs
- 4. “Mainframe”-like RAS on POWER6 drove a huge amount of work that was difficult to attack at the unit level**

# Overview

- 1. Context : POWER5 vs. POWER6 microarchitecture comparison**
- 2. Verification methodology: In the beginning...**
- 3. The times they are a changing: SMT arrives in POWER5**
- 4. POWER6: An in-order design should be simpler, but...**
- 5. Future directions?**

# Future directions

## Predictions

- RAS features will be an increasingly important feature of server systems
  - POWER6 design has set the “bar” to a new high standard to which future processors will have to measure up
    - Power Systems Revenue up 29% in 2Q08 (from 2Q07)
  - Verification methods employed on POWER6 to attack nearly infinite state space created by the combination of SMT and processor recovery features will become standard practice
- A migration of “pre-silicon” verification techniques into “post-silicon” hardware lab verification effort
  - Hardware is the fastest “simulator” available and the state space is getting bigger with SMT